Lesson 11: Enabling Development Tools

Brian Pepin
Developer

.NET Client Team

June 2004

Development Tool Support

- Great tool experience expected by developers
- Tools require support from class library writers to make this happen
  - Microsoft® IntelliSense® hints
  - Help documentation
  - Debug viewers
  - Designer support

- Microsoft Windows® Forms Designer "Whidbey" Features

Demo

Enabling Technologies

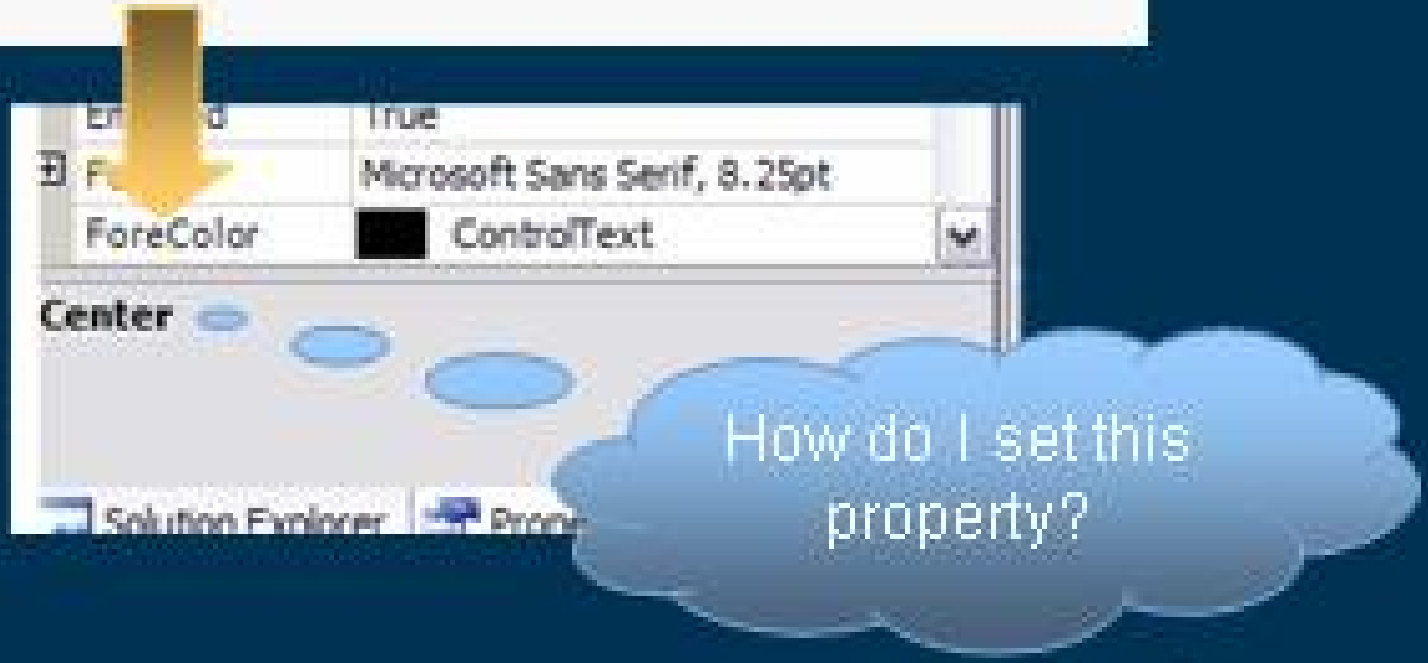
- Metadata attributes / XML Doc Comments
- Object model guidelines
- Custom code to support tools

Metadata Attributes

- **Designers**
  - Description
  - Browsable
  - Bindable
  - ListBindable
  - DefaultValue
- **IntelliSense**
  - EditorBrowsable
  - XML Documentation
- **Debugging**
  - DebuggerDisplay
  - DebuggerBrowsable

Metadata Attributes
Example: Bad Experience

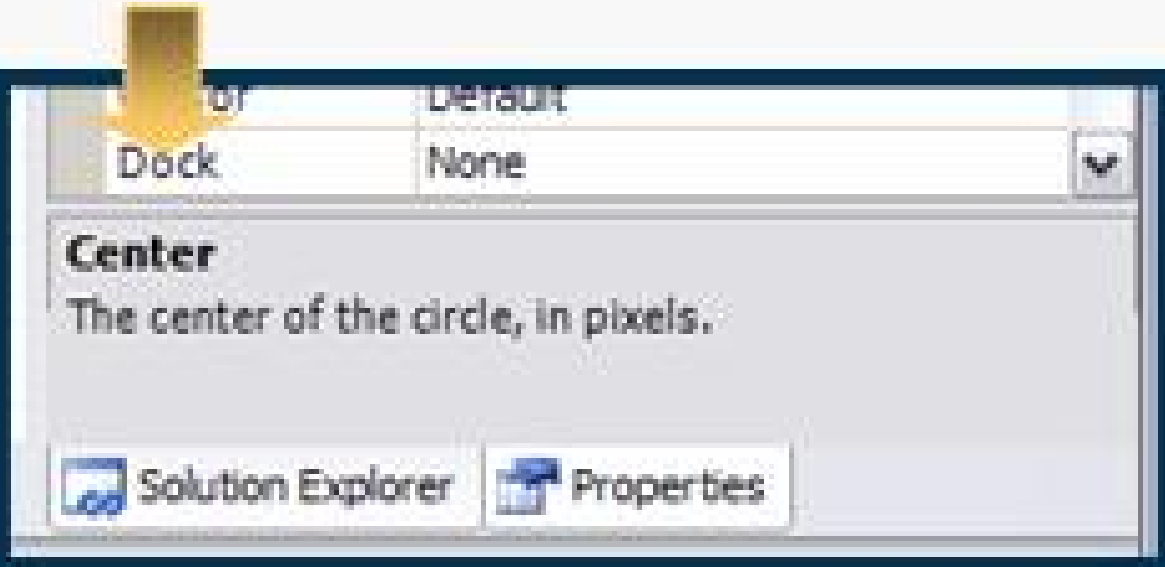
```
public class Circle {
    public Point Center { get; set; }
    public int Radius { get; set; }
}
```



Metadata Attributes

Example: Good Experience

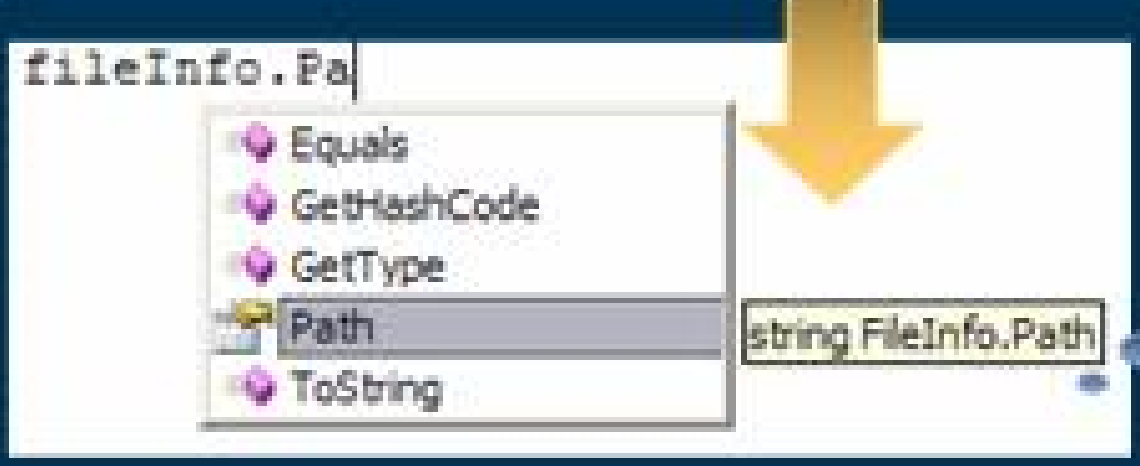
```
public class Circle {
    [Description("The center of the circle.")]
    public Point Center { get; set; }
    public int Radius { get; set; }
}
```



XML Documentation

Example: Bad Experience

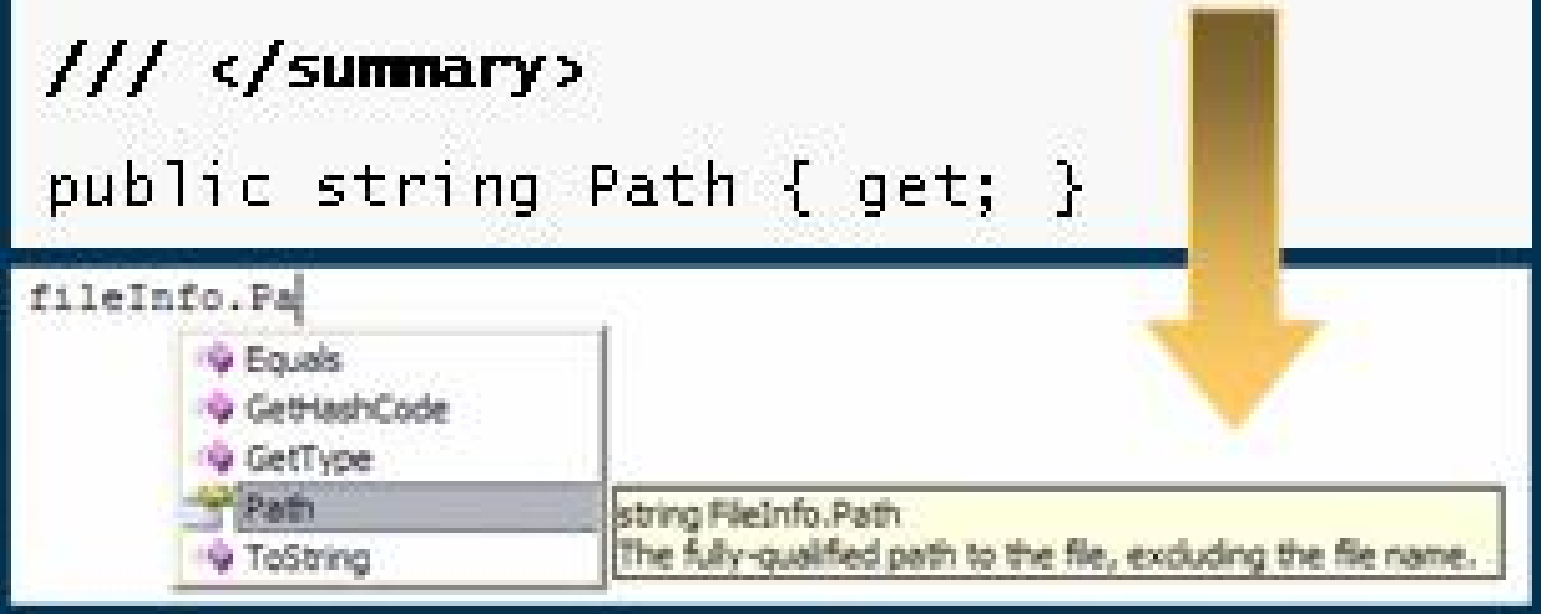
```
public class FileInfo {
    public string Path { get; }
}
```



XML Documentation

Example: Good Experience

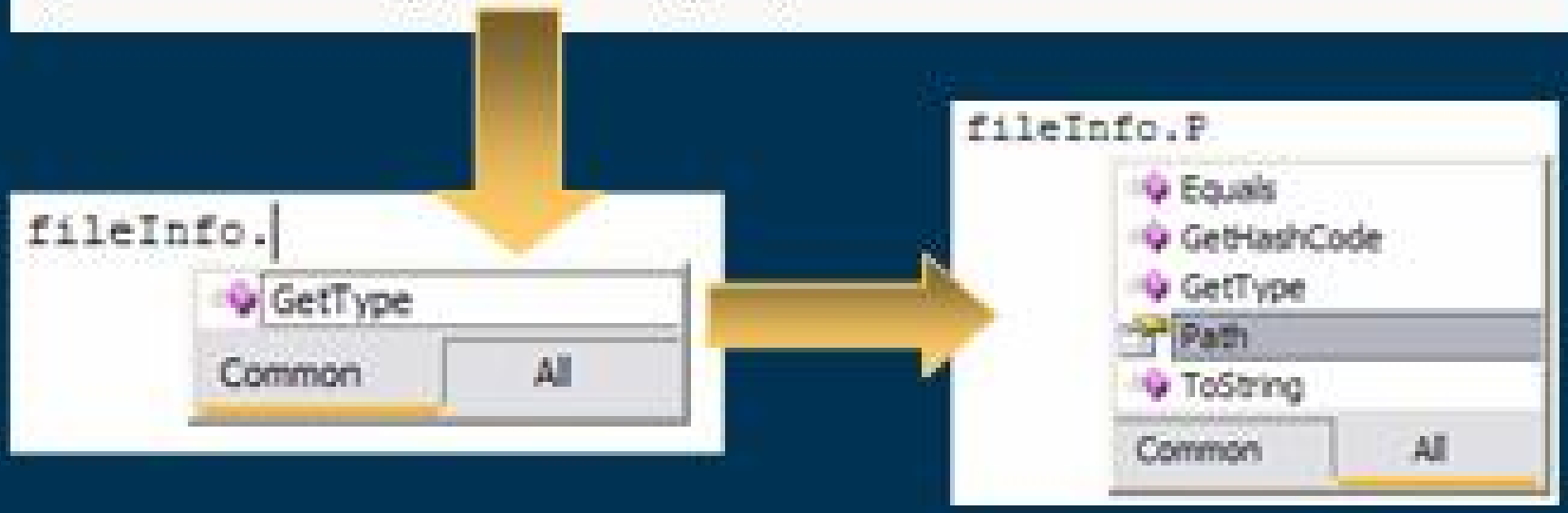
```
/// <summary>
///  The fully-qualified path_
/// </summary>
public string Path { get; }
```



XML Documentation

Example: Filtering

```
/// <filterpriority>2</filterpriority>
[EditorBrowsable(EditorBrowsableState.Advanced)]
public string Path { get; }
```



Object Model Guidelines

- **Object or Component?**
  - Object: lightweight, transient, or utility
  - Component: properties, events, configure and use pattern
- **Properties**
  - No interdependencies
  - Exceptions on setters only
- **Constructors**
  - Components should have an empty ctor

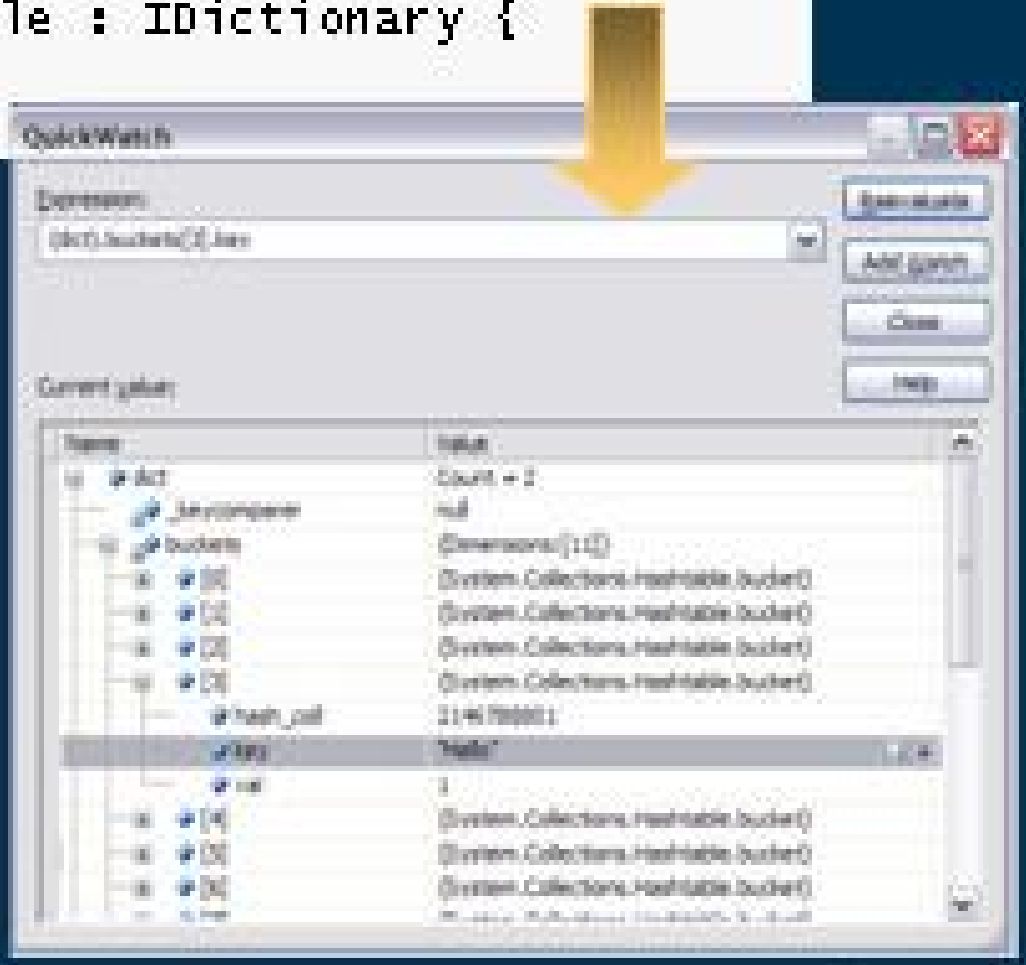
Custom Tool Code

- In COM, you had a flag and a few heuristics
  - Everything was a component
  - Custom data types were minimal
- In Microsoft .NET, you have design-time classes
  - Modify or augment runtime behavior, but don't bloat runtime object
  - Examples: Debugger Type Proxies, Type Converters, Designers

Debugger Type Proxies
Example: Bad Tool Experience

```
public class Hashtable : IDictionary {
}
```

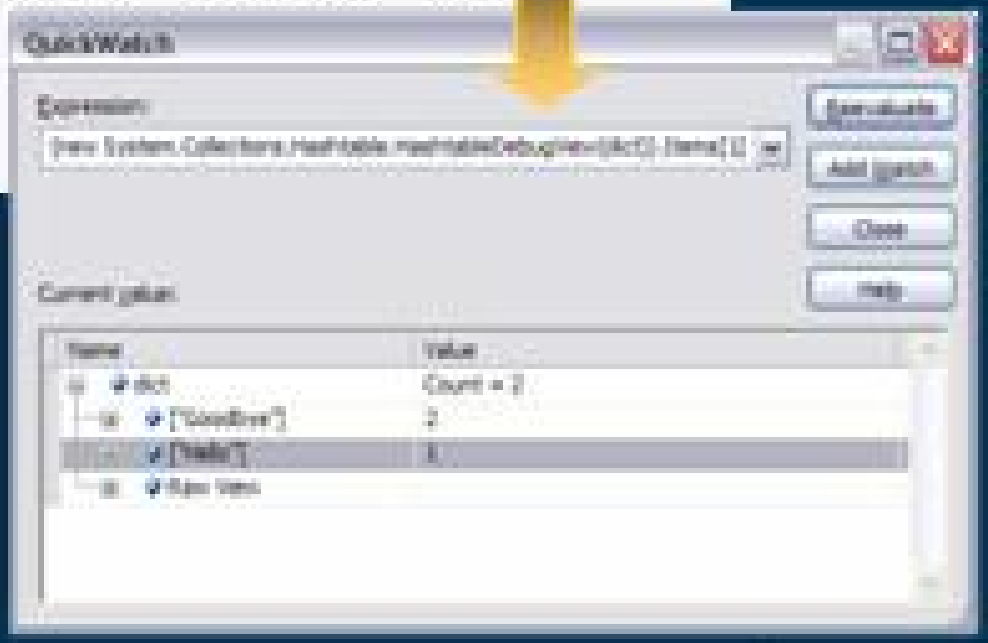
- User plays "find the bucket"
- Irrelevant implementation details exposed



Debugger Type Proxies

Example: Let's Improve Things

```
[DebuggerTypeProxy(typeof(HTDebugView))]
public class Hashtable : IDictionary {
  class HTDebugView {
    public HTDebugView(Hashtable ht) {}
  }
}
```



- What the user expects
- "Raw View" still available

Type Converters

Example: Bad Tool Experience

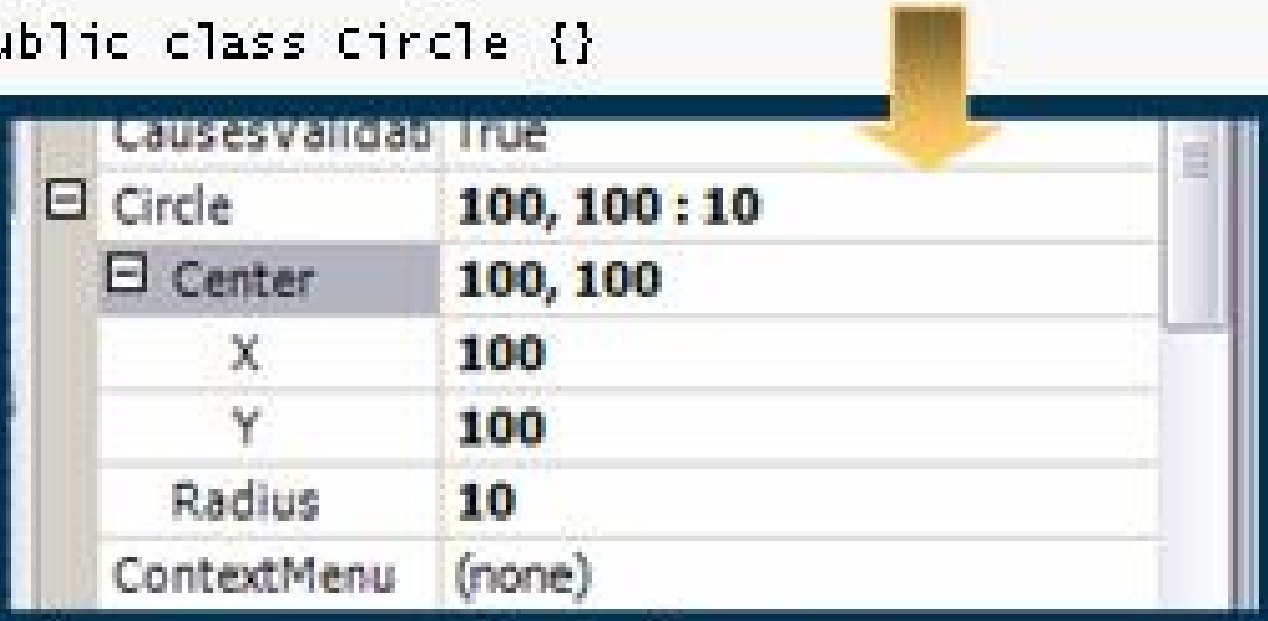
```
public class Circle {
    public Point Center { get; set; }
    public int Radius { get; set; }
}
```

- Can't be edited
- Doesn't display any useful information
- Can't persist to XAML or code

Type Converters

Example: Let's Improve Things

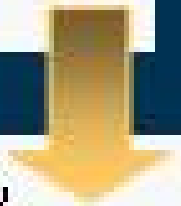
```
[TypeConverter(typeof(CircleConverter))]
public class Circle {}
```



- CircleConverter provides two features
  - Text-to-value and value-to-text conversion
  - Indicates that object exposes properties
- Property Grid and XAML happy, but no code generation yet

Type Converters
Example: Enabling Code Generation

```
Circle c = (Circle)value;
ConstructorInfo ctor = typeof(Circle).
    GetConstructor(new Type[] {
        typeof(Point),
        typeof(int) });
return new InstanceDescriptor(ctor,
    new object[] { c.Center, c.Radius });
```



```
this.myControl1.Circle = new Circle(new
Point(100, 100), 10);
```

Task

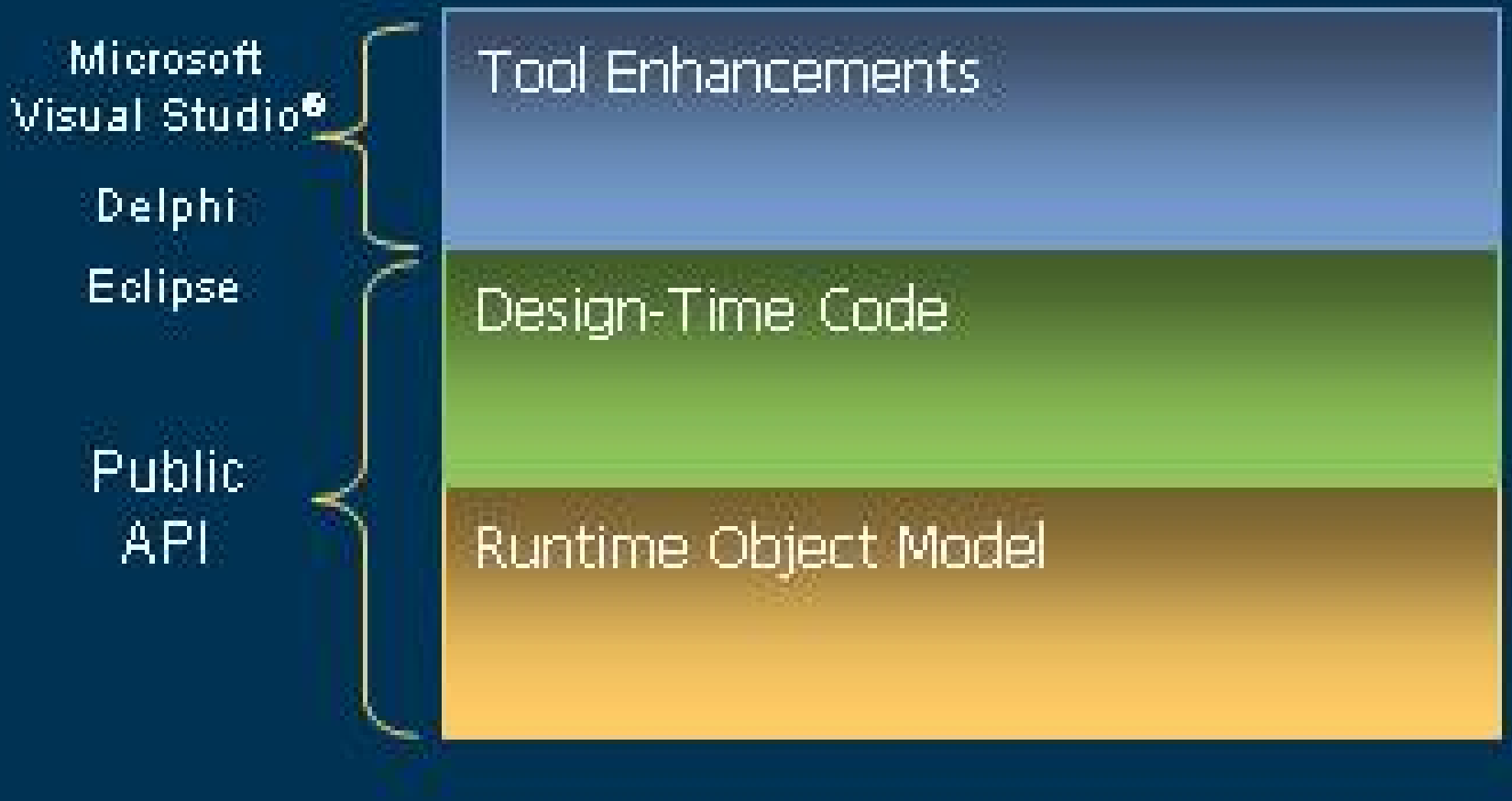
Writing a TypeConverter

- **Priority 1**
  - Bi-directional string-to-value conversions
  - ConvertTo InstanceDescriptor for code
- **Priority 2**
  - Provide expandable properties
  - Provide standard values for selections
- **Priority 3**
  - Bi-directional byte[] for binary blobs

Designers
What Are They?

- Class that is instantiated by the tool to change the behavior of a run-time object
- Defaults provided for common base classes
  - IComponent
  - Control
  - FrameworkElement
- Same architecture, but different models for each ASP.NET, Windows Forms, "Avalon"

Tool Extensibility



Lesson 11 Summary

- Developers expect a rich tool experience
- Enabling this experience requires work from API developers
- All public APIs are candidates, so include this in your schedule